Deep Learning for Computer Vision: **Practical Training**

Shai Bagon

WAIC


My Neural Net
SGD tricks
Regularization
BatchNorm
Initialization
...

Agenda

- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization

[redacted]

Gradient Descent

$$\mathcal{L}(\theta; \{(\boldsymbol{x}_i, y_i)\}_{i\in 1..N}) = \frac{1}{|N|}\sum\nolimits_{i\in\{1..N\}} \mathcal{L}(f(x_i;\theta), y_i)$$

Stochastic Gradient Descent (SGD)

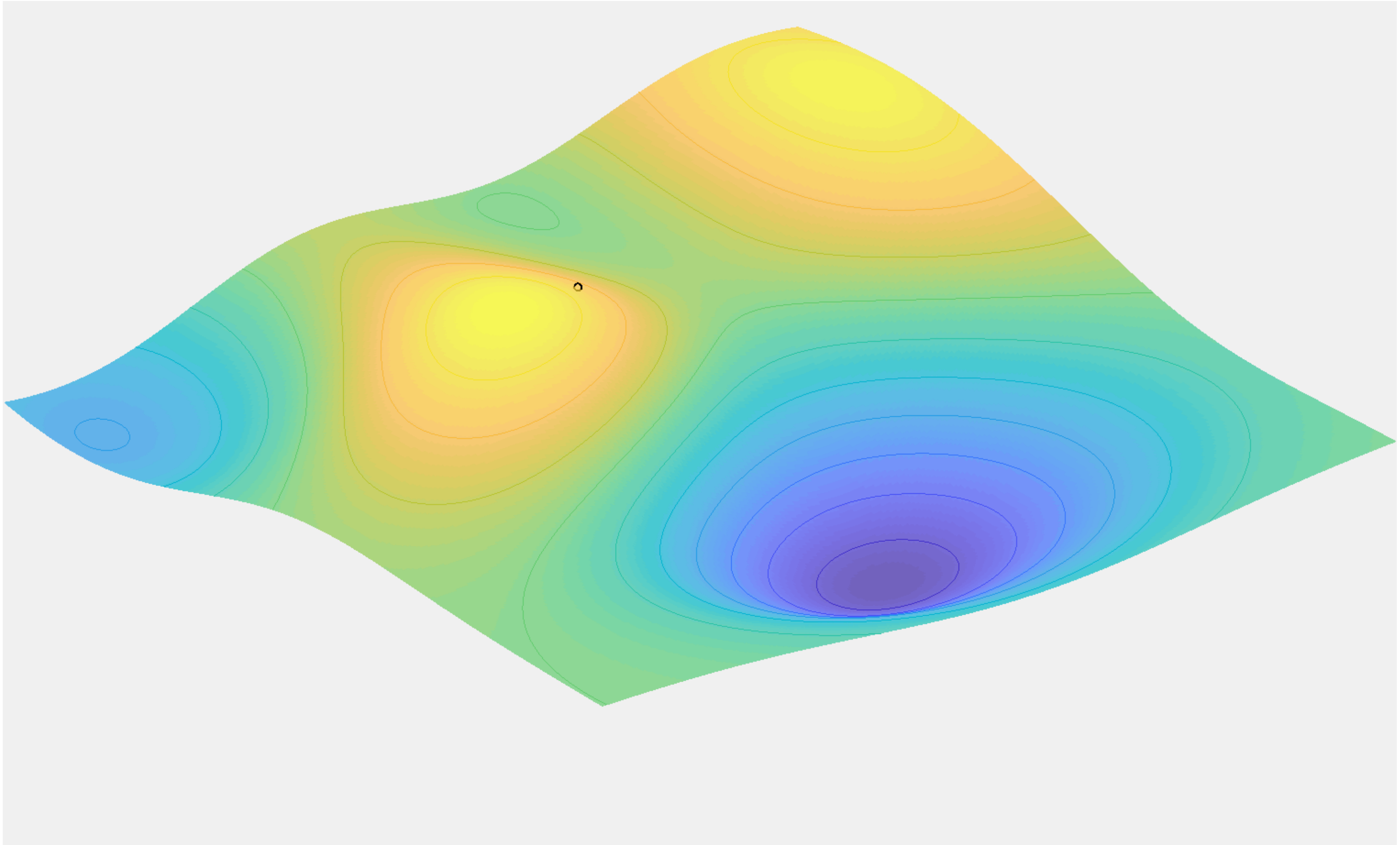
$$\mathcal{L}(\theta;\{(\boldsymbol{x}_i, y_i)\}_{i\in 1..N}) \approx \frac{1}{|B|}\sum_{i\in\{B\}} \mathcal{L}(f(x_i;\theta), y_i)$$

Update rule:

$$\theta^{t+1} = \theta^{t} - \eta\nabla\,\mathcal{L}\left(\theta^{t};\{(\boldsymbol{x}_i, y_i)\}_{i\in\{B\}}\right)$$

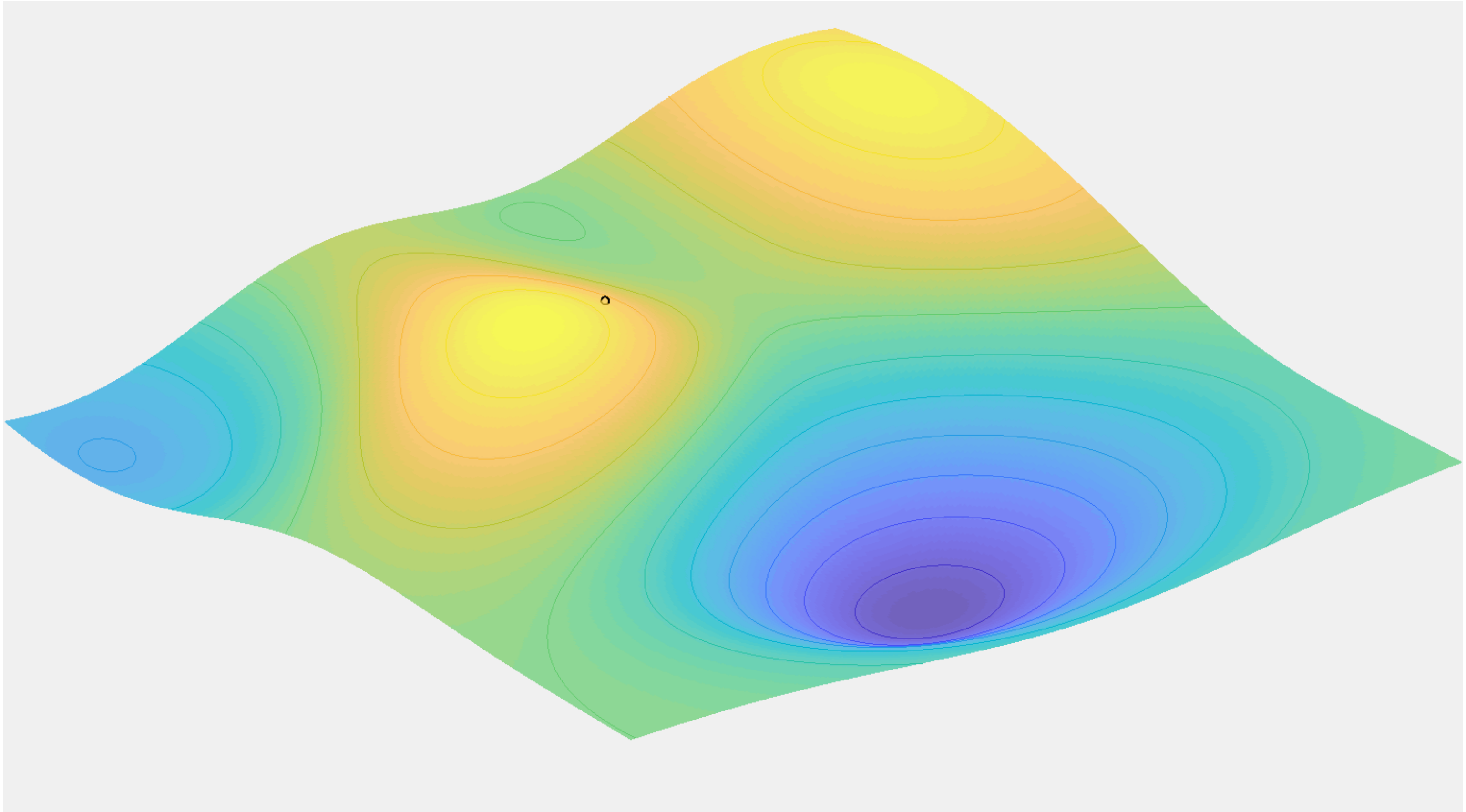
Vanilla SGD

$$\theta^{t+1} = \theta^{t} - \boldsymbol{\eta}\nabla\, \mathcal{L}(\theta^{t})$$



Vanilla SGD

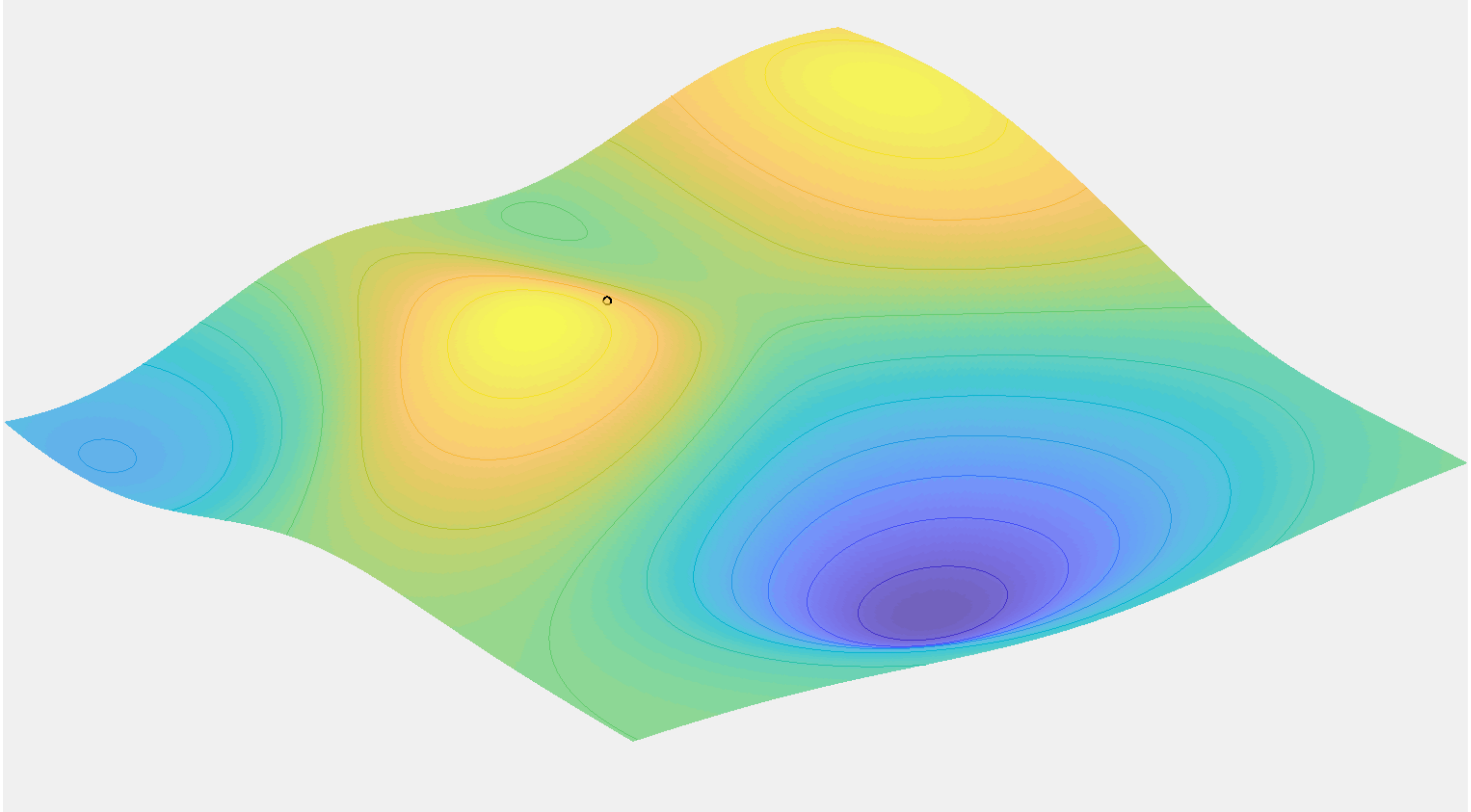
$$\theta^{t+1} = \theta^{t} - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^{t})$$



High learning rate

Vanilla SGD

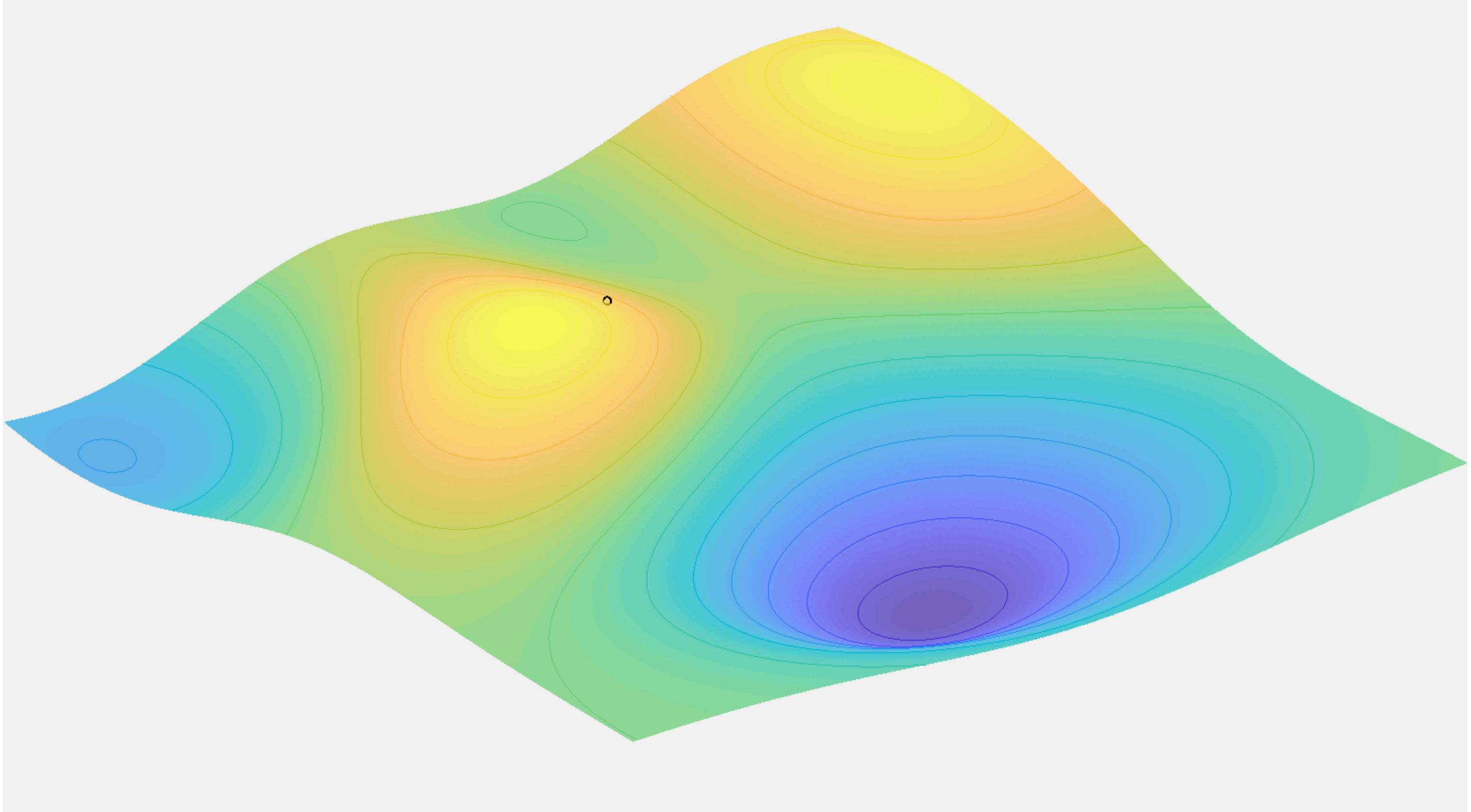
$$\theta^{t+1} = \theta^{t} - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^{t})$$



Low learning rate

Vanilla SGD

$$\theta^{t+1} = \theta^{t} - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^{t})$$



“Ideal” learning rate

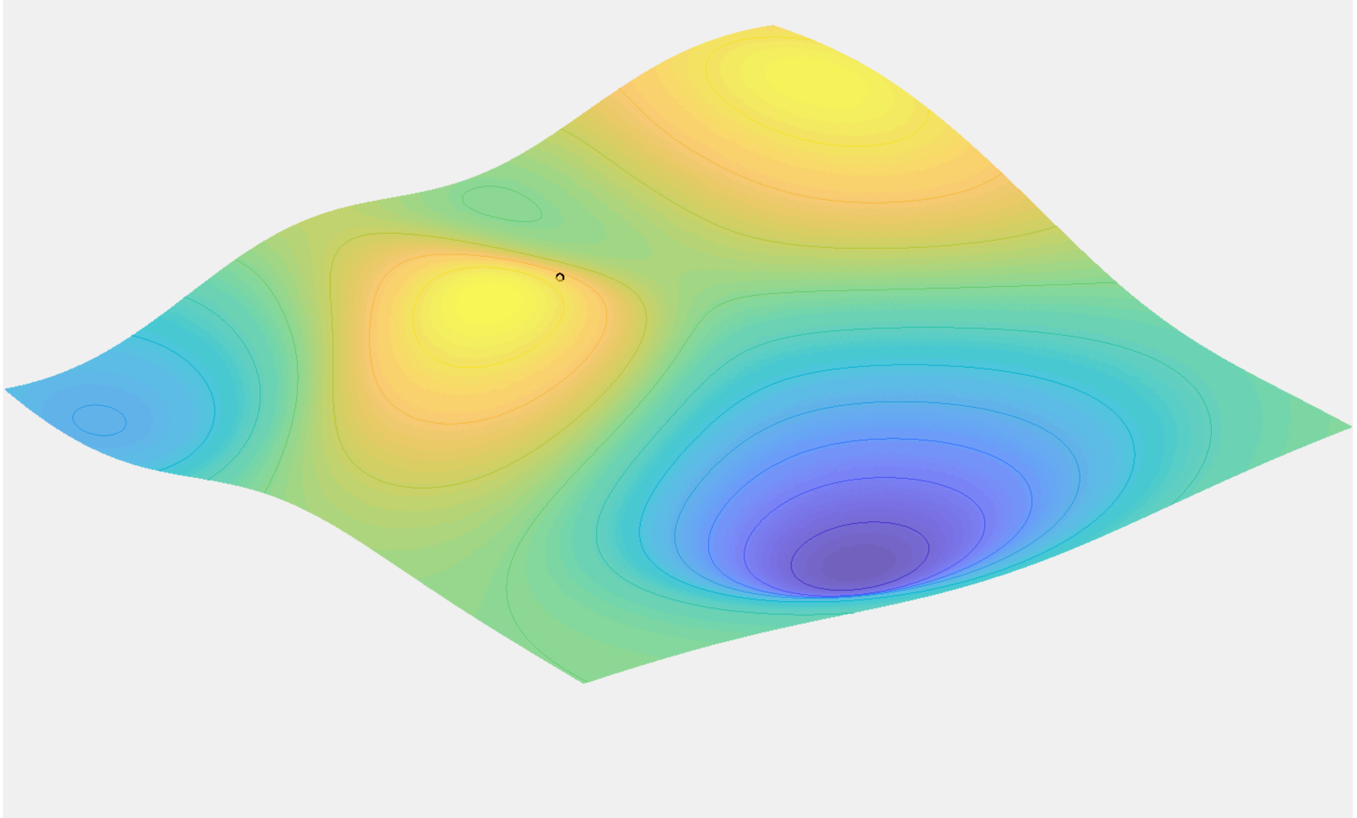
Vanilla SGD

$$\theta^{t+1} = \theta^{t} - \boldsymbol{\eta}\nabla\, \mathcal{L}(\theta^{t})$$

SGD with Momentum

$$\theta^{t+1} = \theta^{t} + \boldsymbol{\mu} v^{t} - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^{t})$$

$$v^{t+1} = \mu v^{t} - \eta \nabla \mathcal{L}(\theta^{t})$$



[redacted]

SGD with Momentum

$$\theta^{t+1} = \theta^{t} + \boldsymbol{\mu} v^{t} - \boldsymbol{\eta} \nabla \mathcal{L}(\theta^{t})$$

$$v^{t+1} = \mu v^{t} - \eta \nabla \mathcal{L}(\theta^{t})$$

Adam

1st moment (like SGD)

$$v^{t+1} = \beta_1 v^t + (1 - \beta_1)\nabla\mathcal{L}(\theta^t) \Big/ (1 - (\beta_1)^t)$$

Unbiased estimation

2nd moment ("variance")

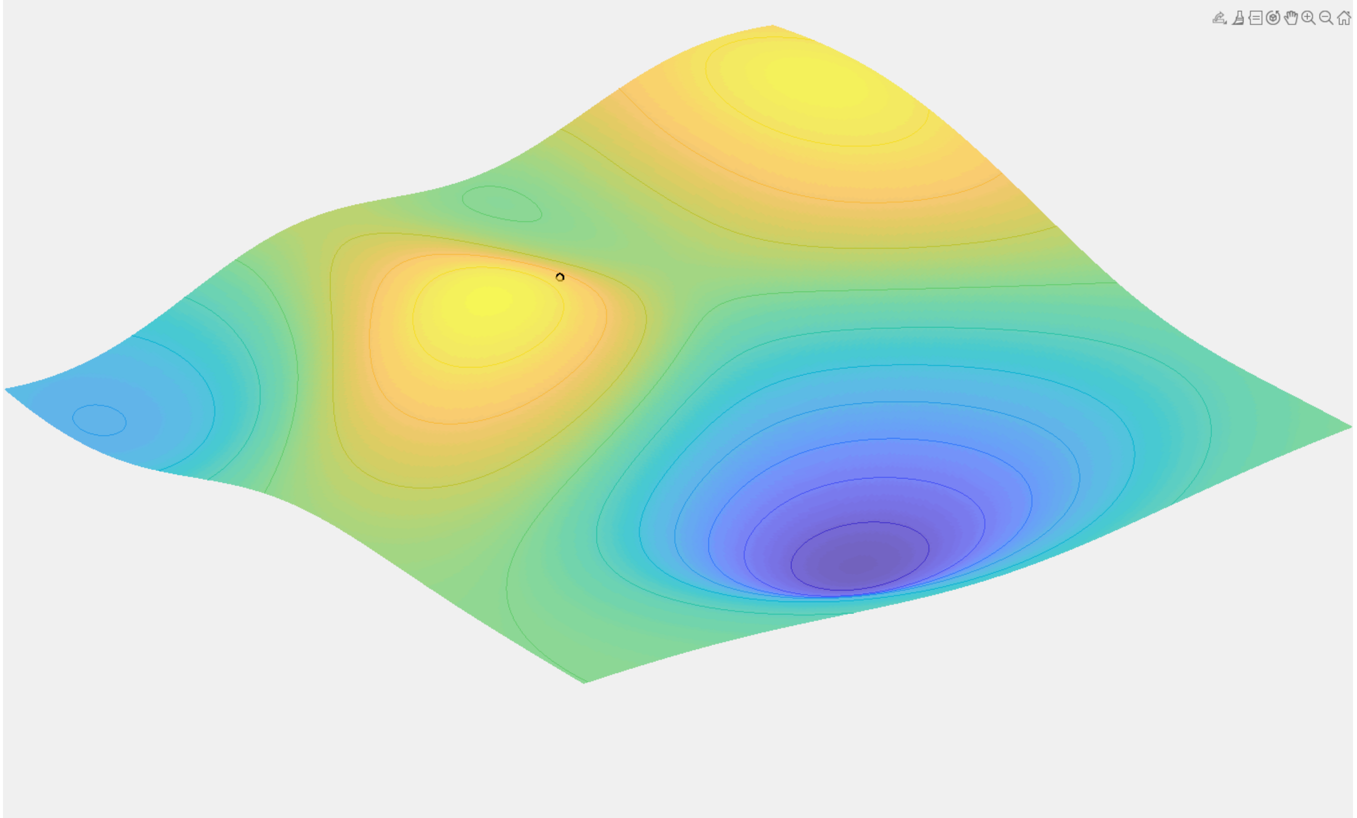
$$m^{t+1} = \beta_2 m^t + (1 - \beta_2)\nabla\mathcal{L}(\theta^t)^2 \Big/ (1 - (\beta_2)^t)$$

Weight the change by the variance

$$\theta^{t+1} = \theta^t - \eta\left(v^{t+1} \Big/ \left(\sqrt{m^{t+1}} + \varepsilon\right)\right)$$

Adam

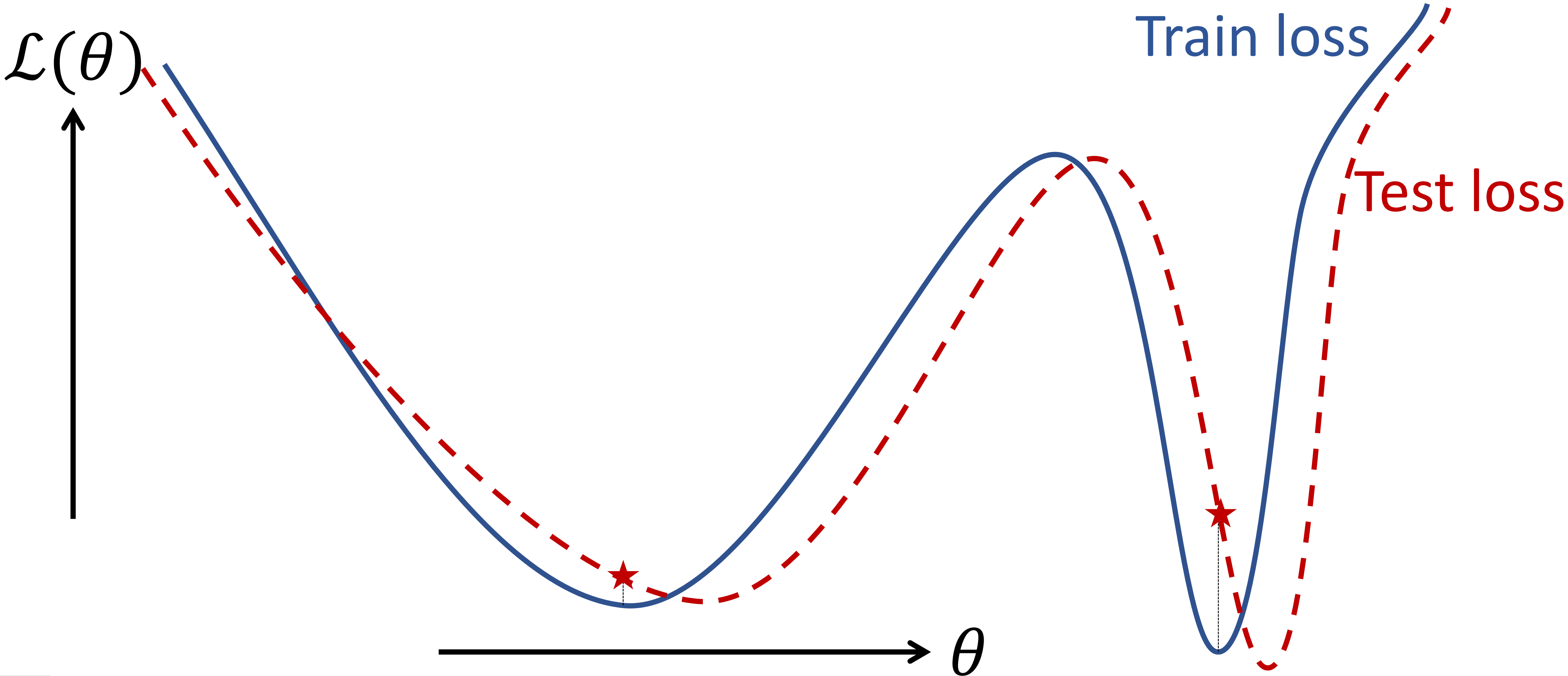
$$\theta^{t+1} = \theta^{t} - \eta \cdot \ v^{t+1} \Big/ \left(\sqrt{m^{t+1}} + \varepsilon\right)$$



Other SGD Update Rules

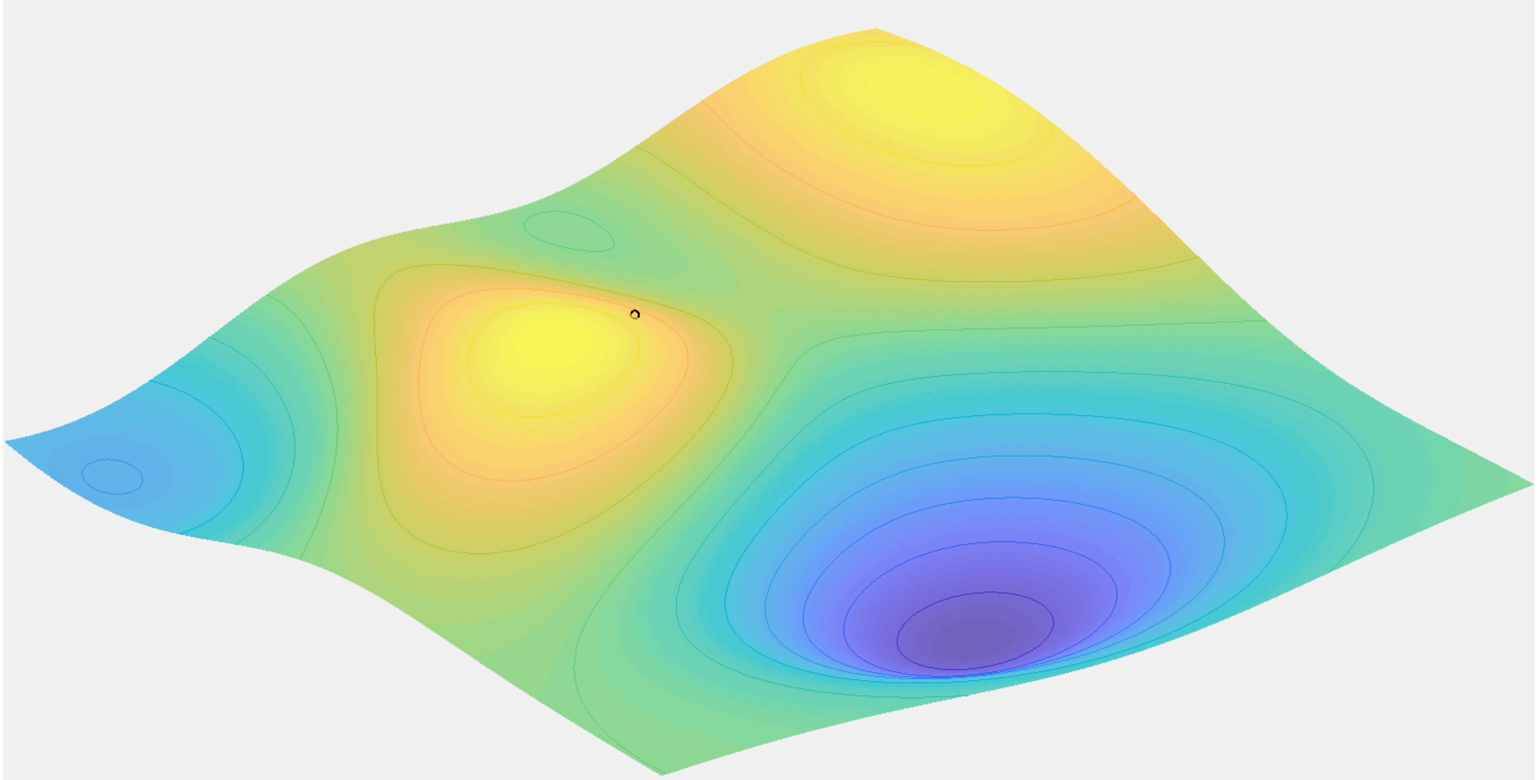
A good review can be found here.

Do we want fast convergence?



Vanilla SGD

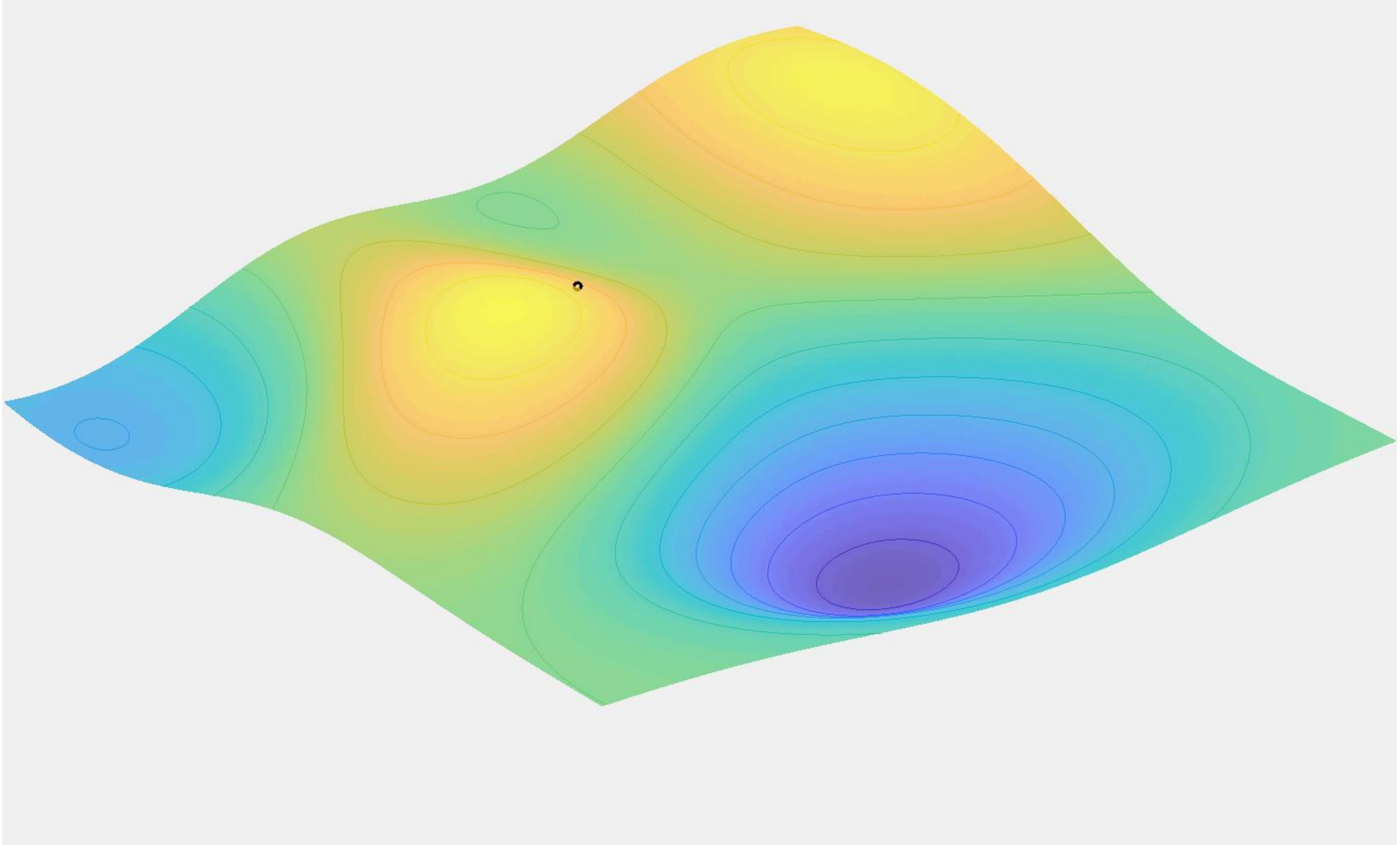
$$\theta^{t+1} = \theta^{t} - \boldsymbol{\eta}\nabla\,\mathcal{L}(\theta^{t})$$



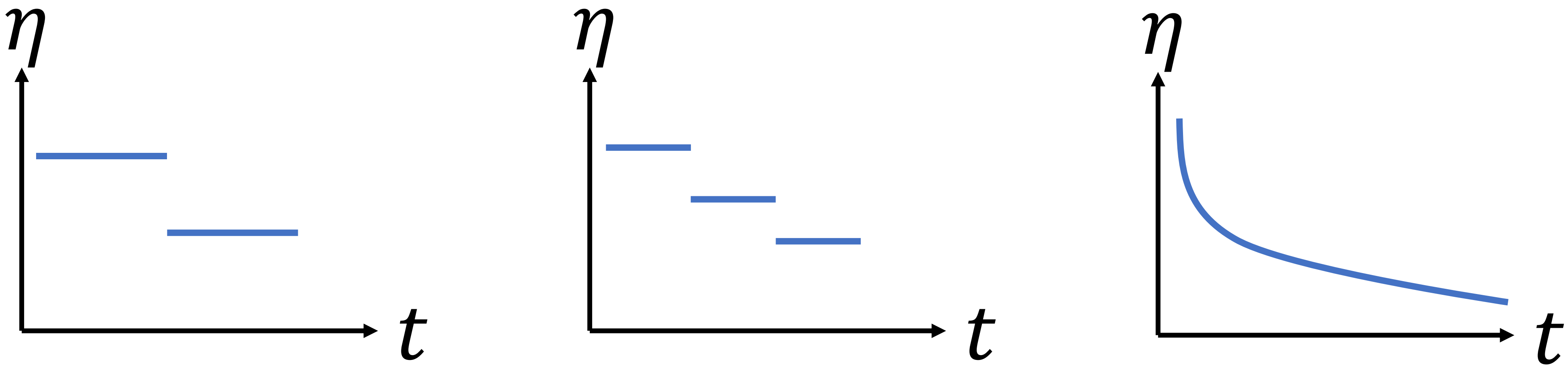
High learning rate

Learning Rate Decay

$$\theta^{t+1} = \theta^{t} - \boldsymbol{\eta_t} \nabla \mathcal{L}(\theta^{t})$$



Learning Rate Decay



SGD - Remarks

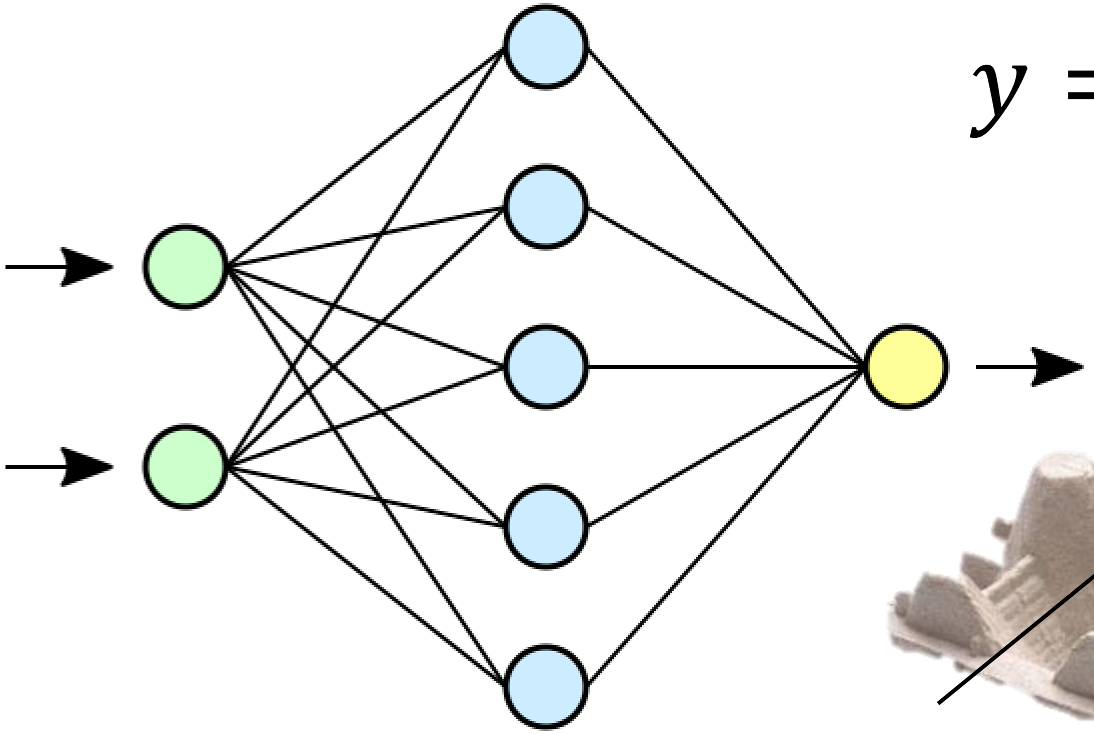
- Guarantees?

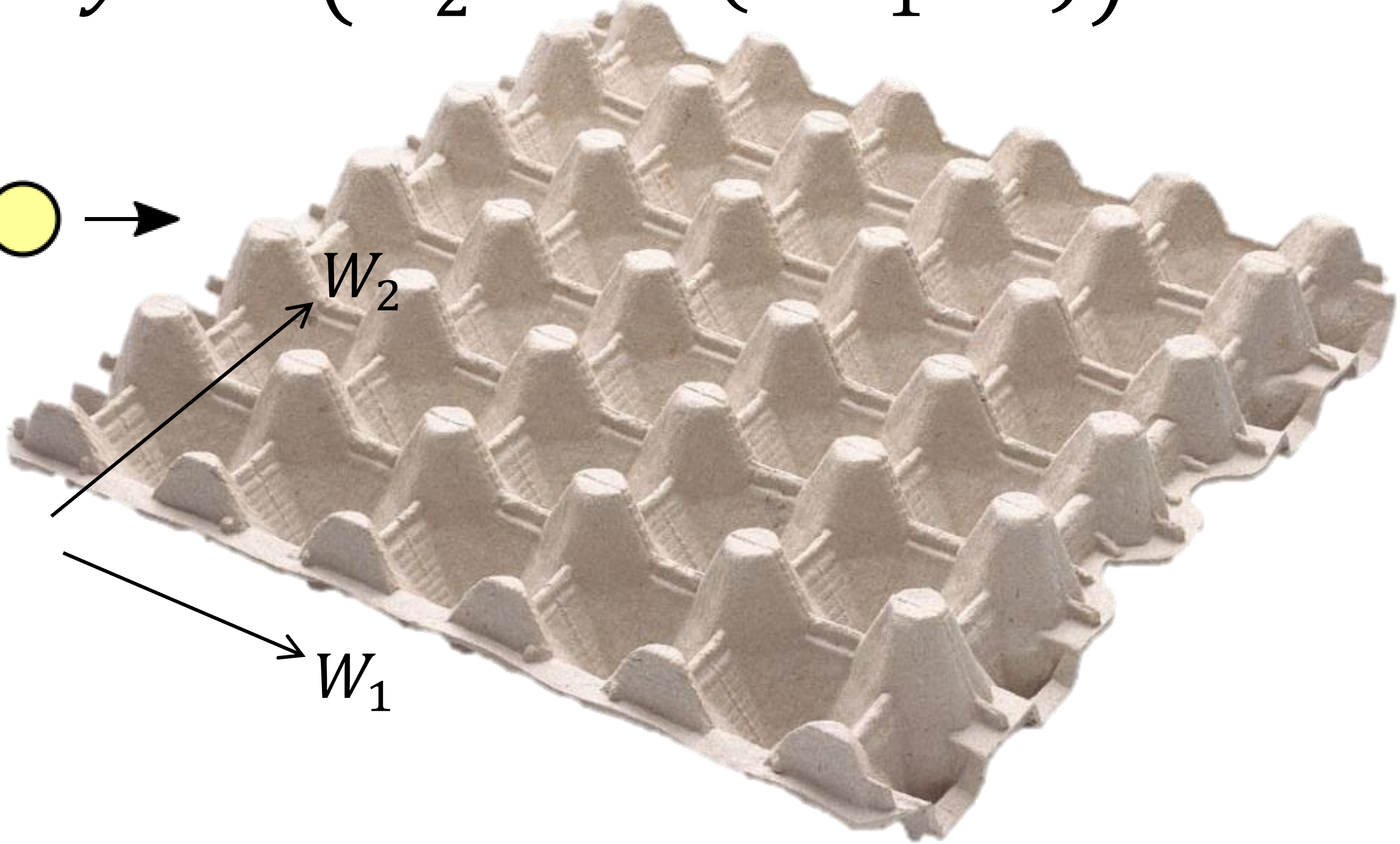
    No!

SGD - Remarks

$$y = \sigma\big(W_2 \cdot \sigma(W_1 \cdot x)\big)$$

$$y = \sigma\big(W_2 P^T \cdot \sigma(P W_1 \cdot x)\big)$$





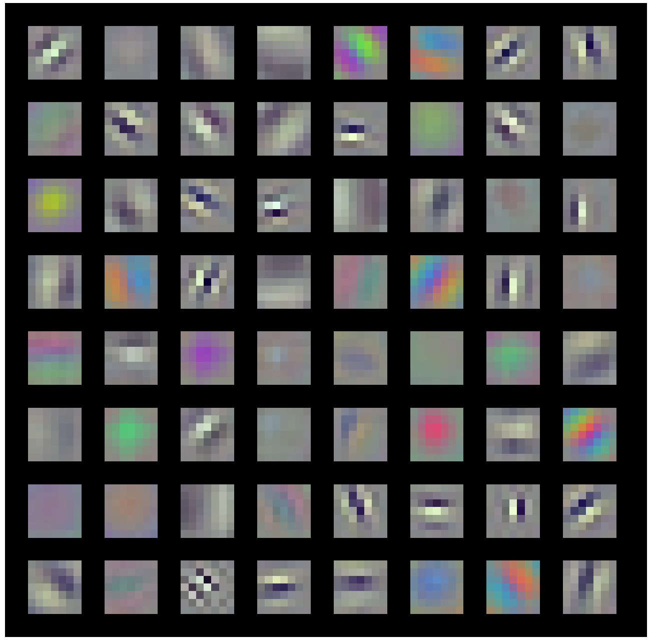
SGD – Starting point

What if we init $\boldsymbol{\theta} = 0$?

$$y = W^T x$$

$$\frac{\partial y}{\partial x} = W$$

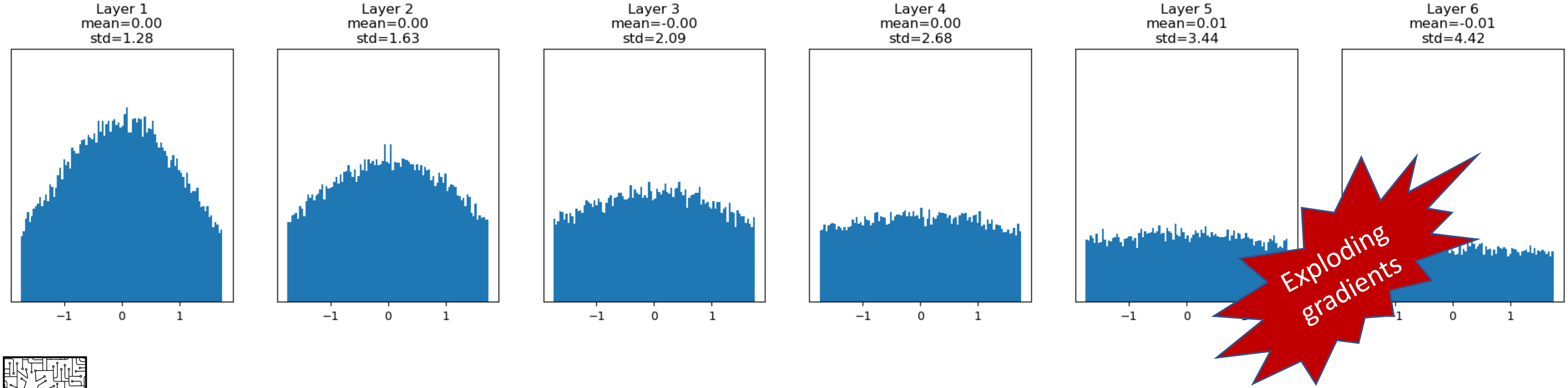
$$\frac{\partial y}{\partial W} = x^T$$



SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

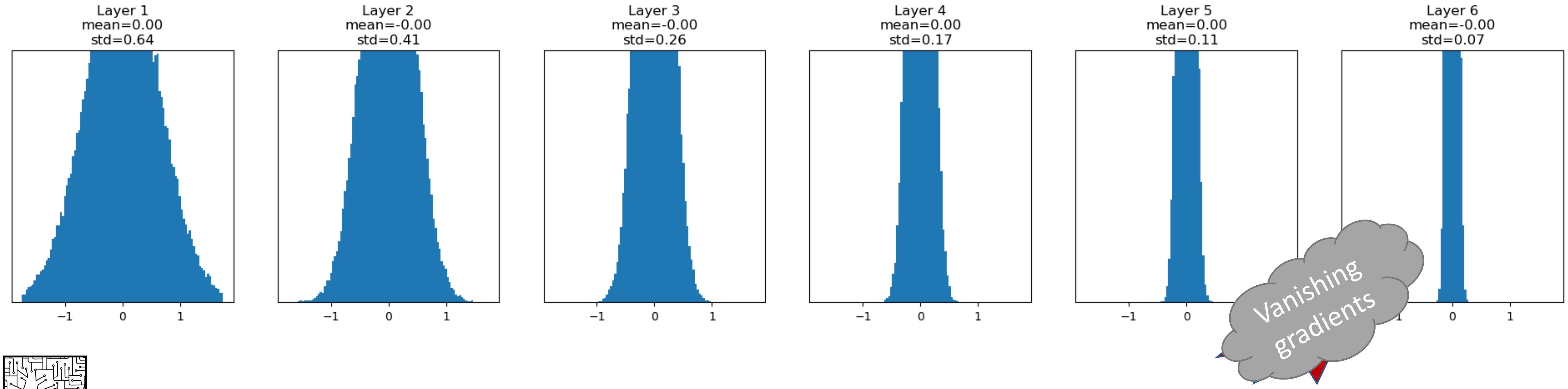
$w_{ij} \sim N(0, 0.02^2)$



SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

$w_{ij} \sim N(0, 0.01^2)$



SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

$$y = W^T x$$

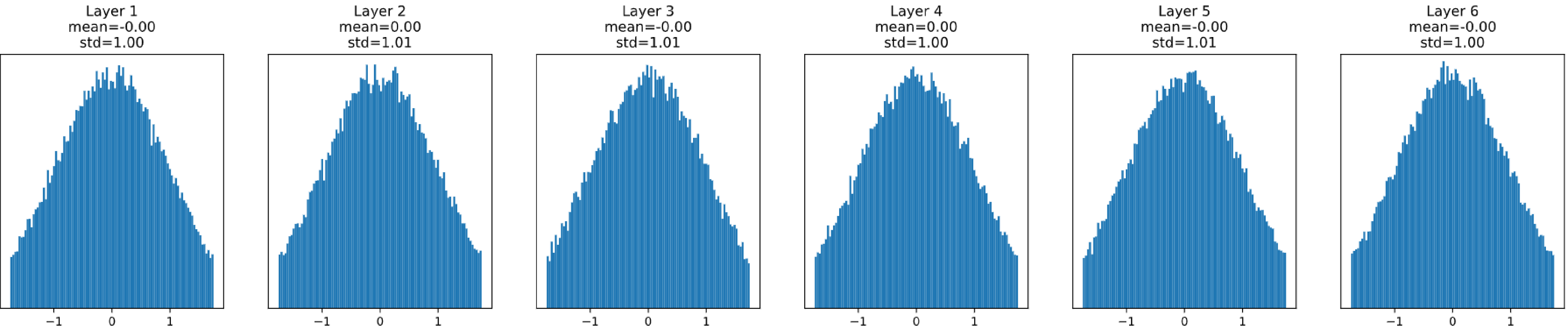
$$\mathrm{var}(y_i) \quad = \quad \mathrm{var}\left(W_i^T x\right)$$

$$\mathrm{var}(w) = {}^{1}/_{D_{in}} \rightarrow \sigma = \sqrt{{}^{1}/_{D_{in}}}$$

SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

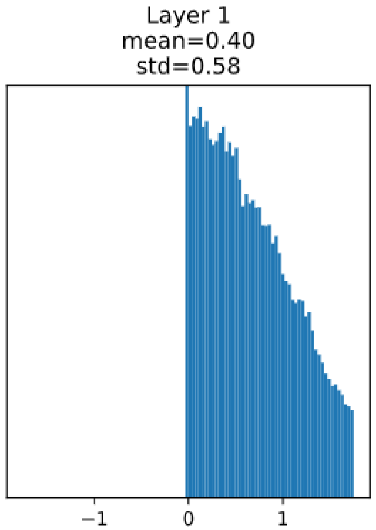
$$w_{ij} \sim N\left(0, {}^{1}/_{D_{in}}\right)$$



SGD – Starting point

MLP, 6 layers, hidden dim=4096, **ReLU** activation

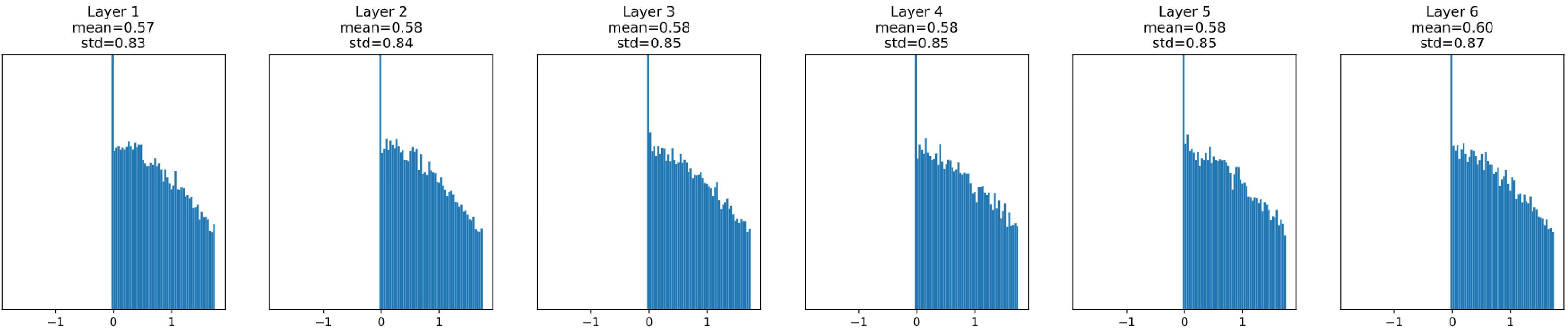
$$w_{ij} \sim N\left(0, {}^{1}/_{D_{in}}\right)$$



SGD – Starting point

MLP, 6 layers, hidden dim=4096, **ReLU** activation

$$w_{ij} \sim N\left(0, \mathbf{2}/_{D_{in}}\right)$$



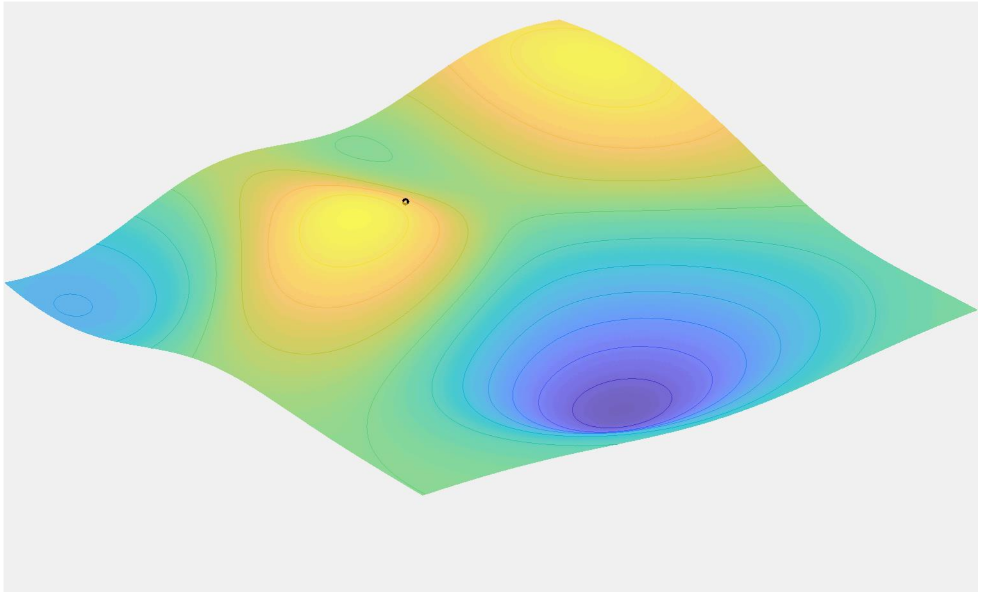
jupyter notebook is available

SGD – Starting point

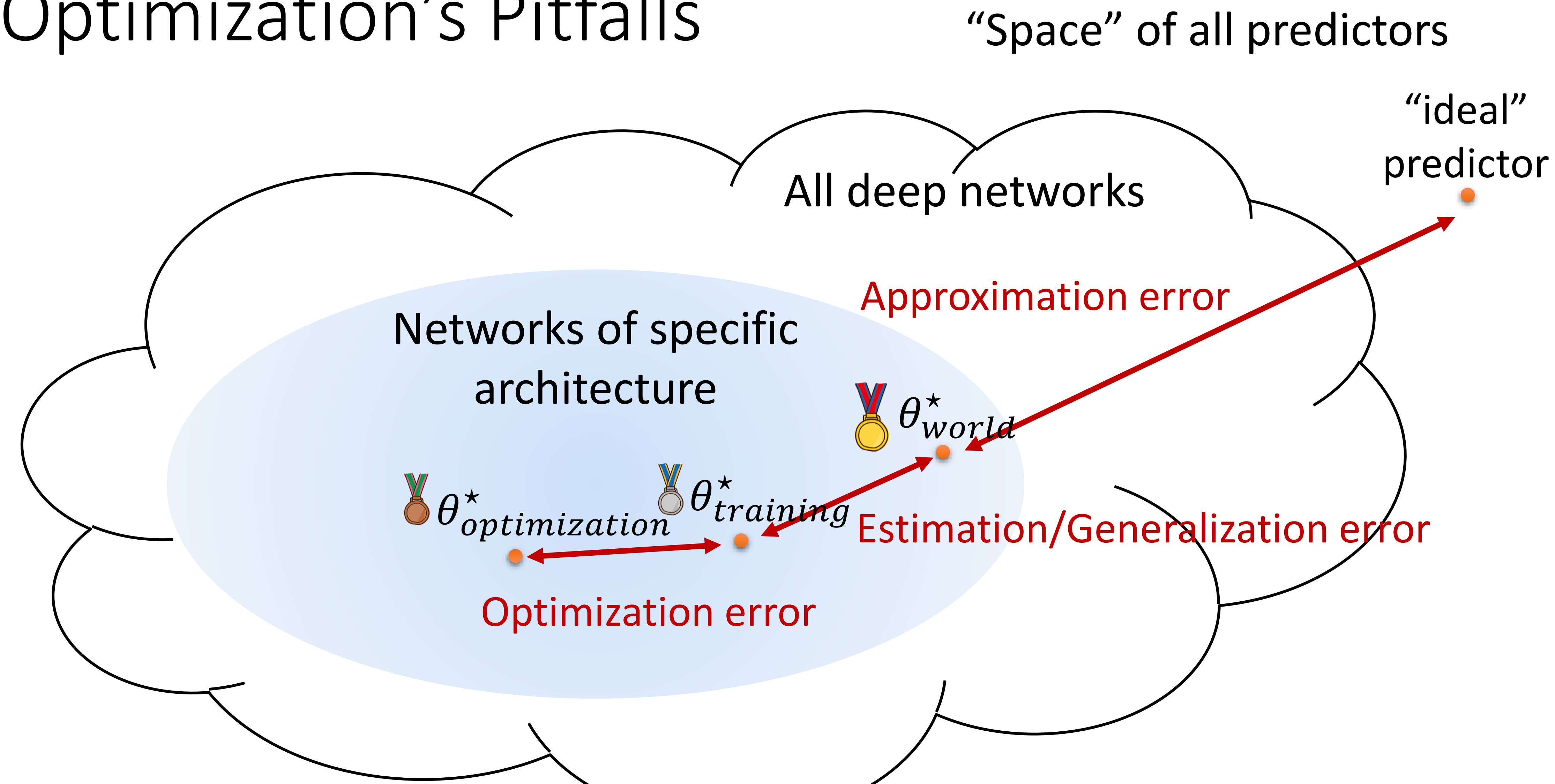
- Where we start the SGD is crucial
- Init depends on the architecture
- Xavier/Kaiming can be easily extended to Conv layers
- Inputs should be normalized to $\sim N(0, 1)$

Agenda

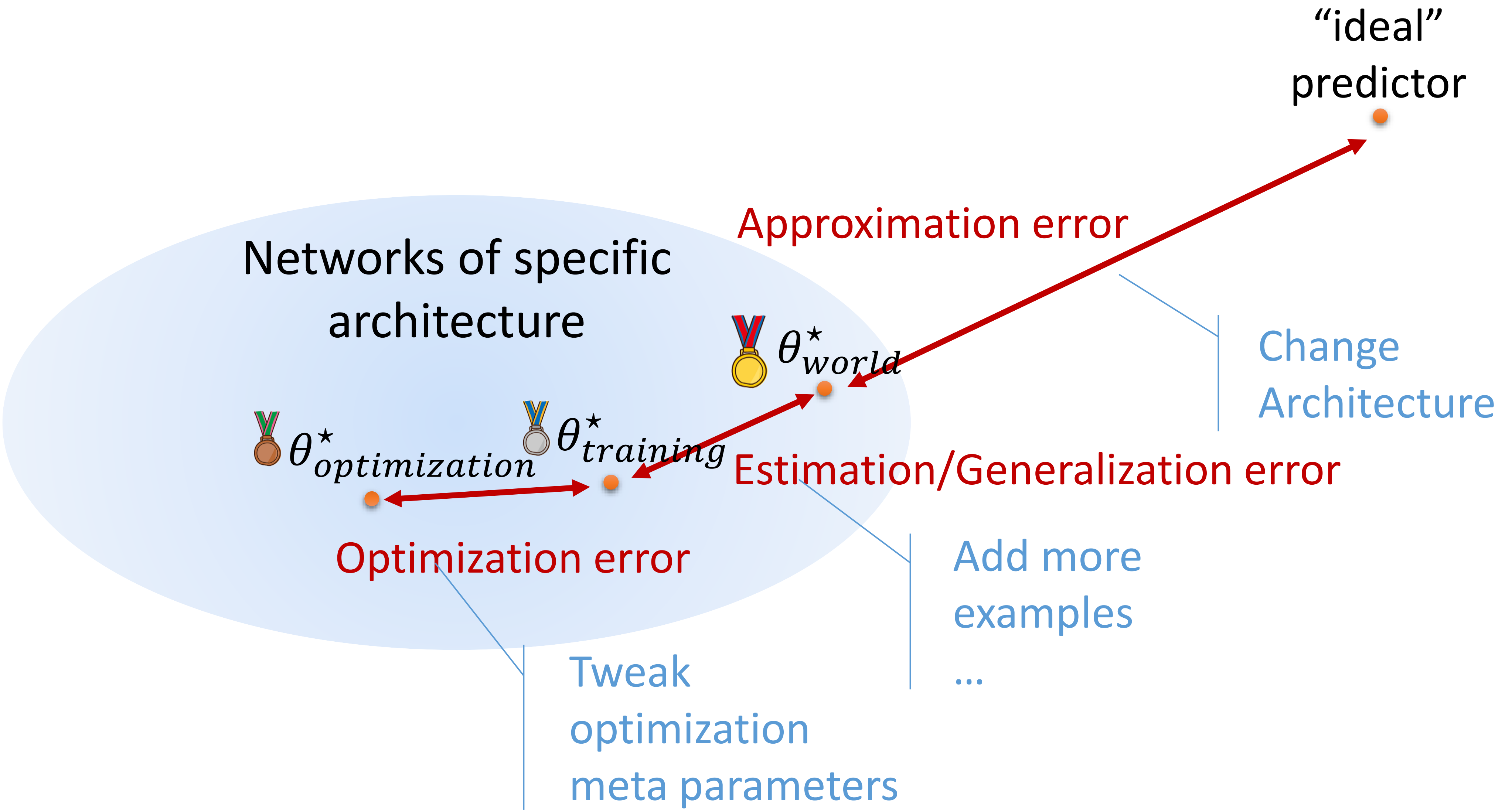
- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization



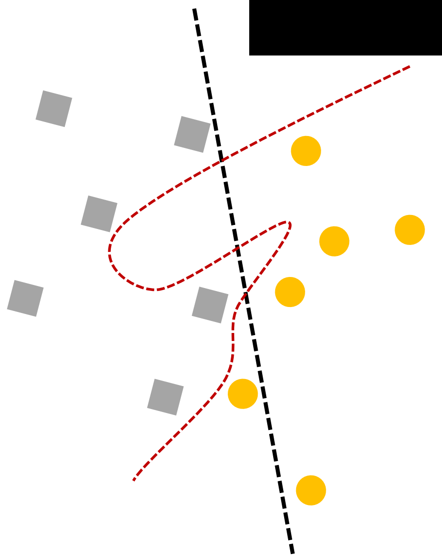
Optimization's Pitfalls



Optimization’s Pitfalls

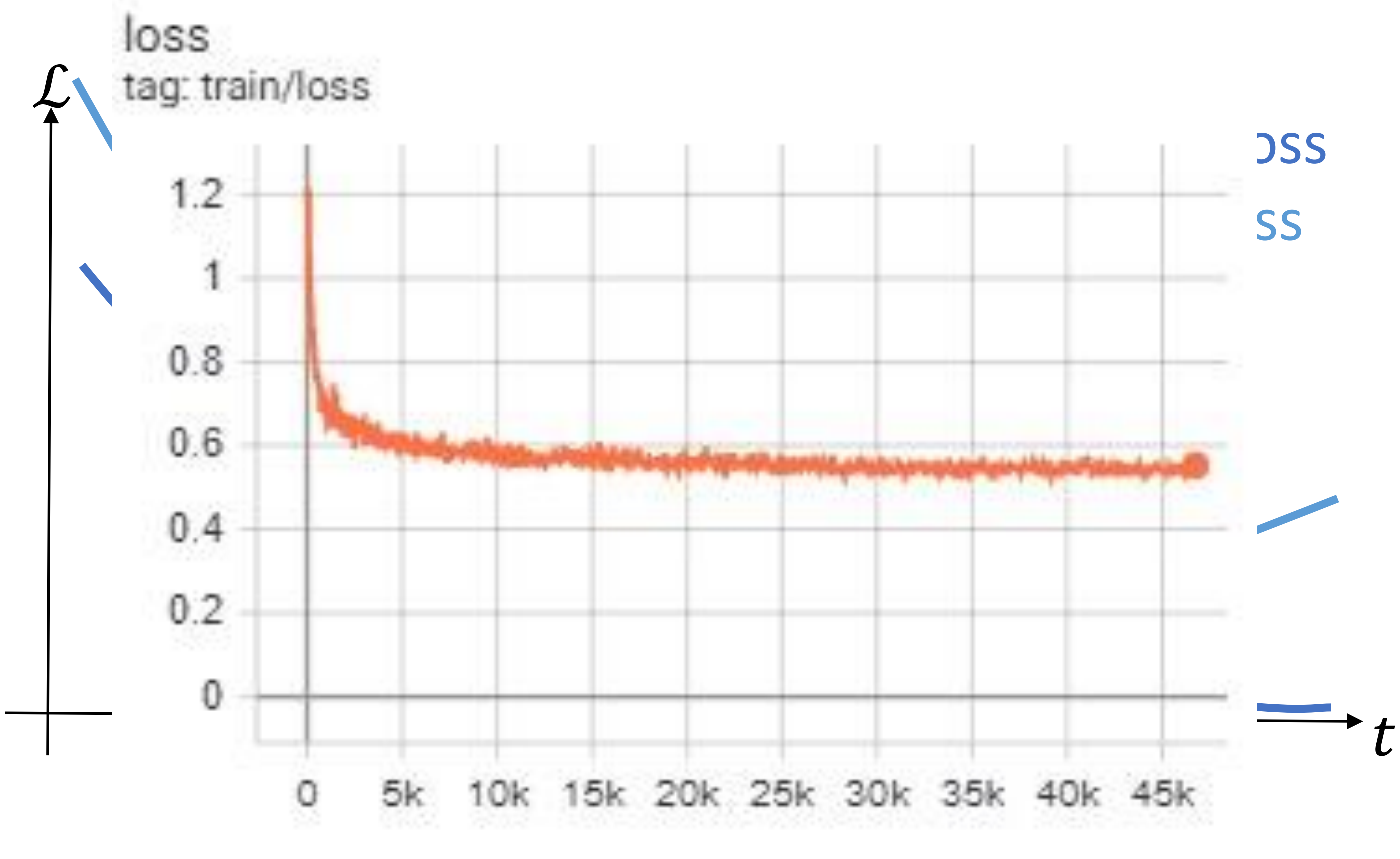


Optimization’s Pitfalls



- How to look at the loss-vs-iterations for train/test set?

- Overfitting



Regularization

$$\mathcal{L}(\theta; \boldsymbol{X}) = CE(\theta; \boldsymbol{X}) + \lambda \|\theta\|_p$$

Regularization

- Data augmentation

Torchvision: `transforms`
Albumenations: https://github.com/albumentations-team/albumentations

[redacted]

Regularization

Dropout

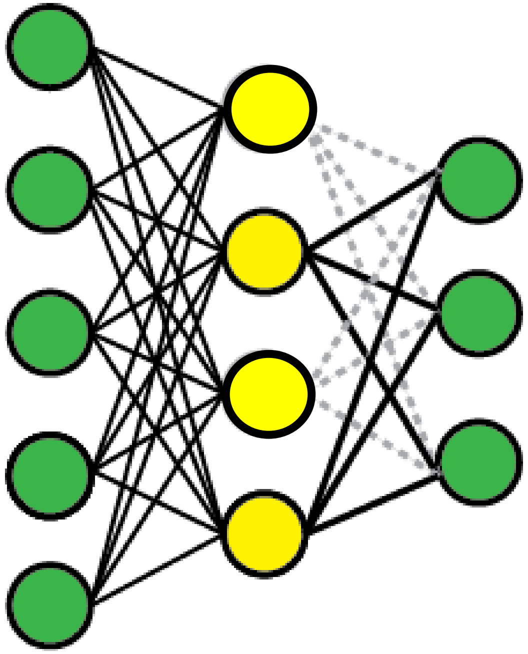
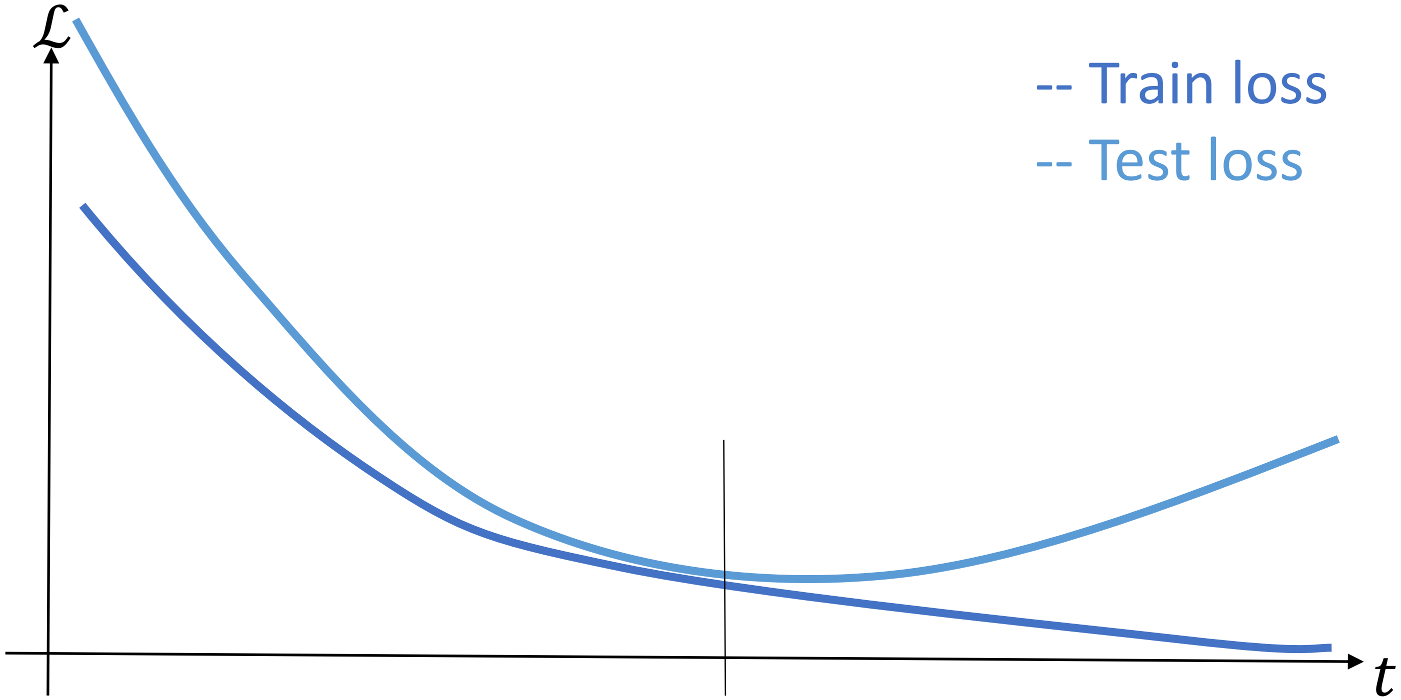


Image source

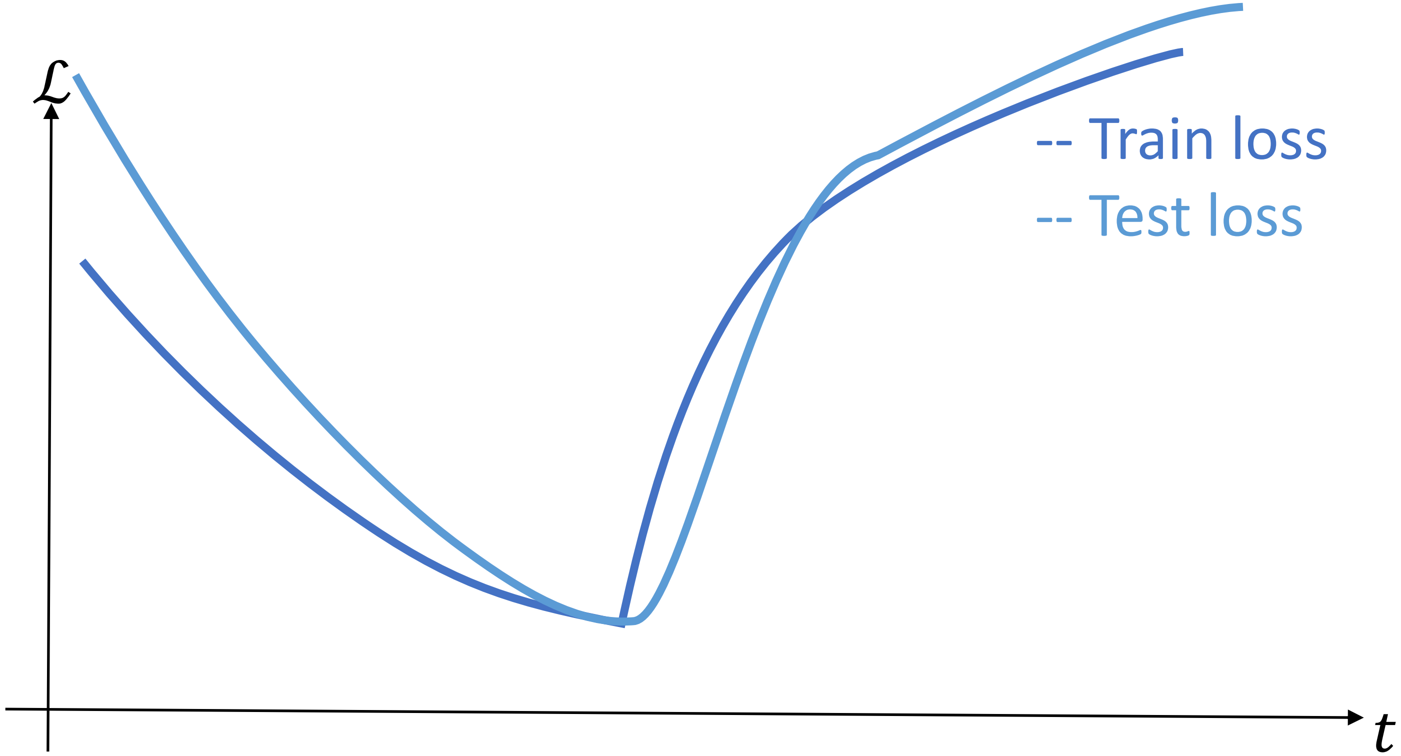
Regularization

Early Stopping



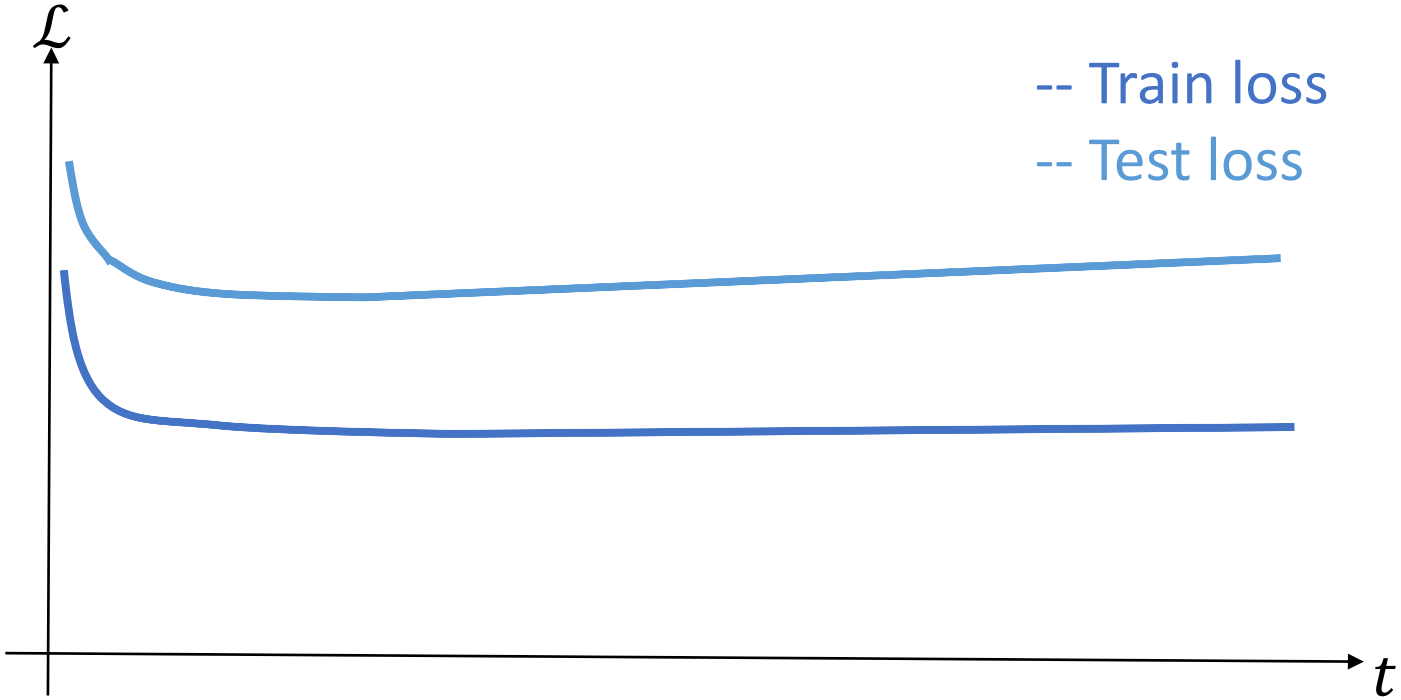
Optimization's Pitfalls

Exploding gradients



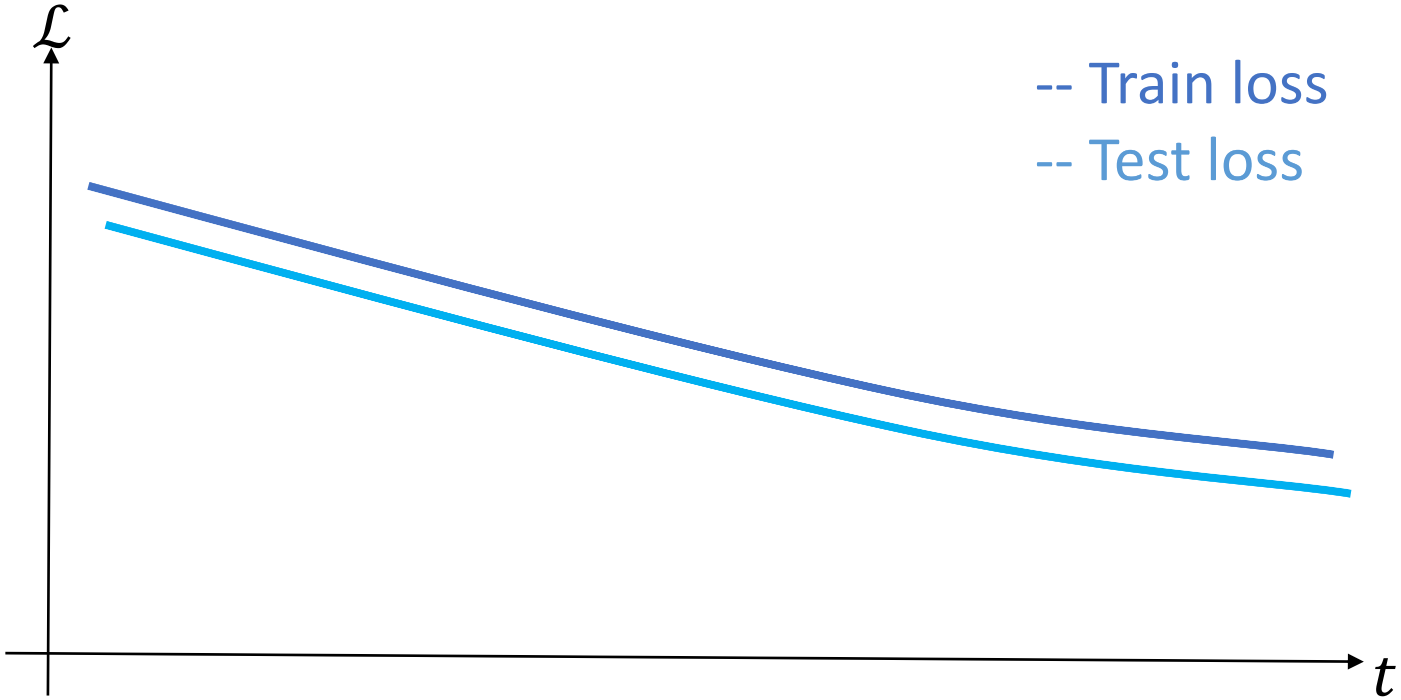
Optimization's Pitfalls

Vanishing gradients



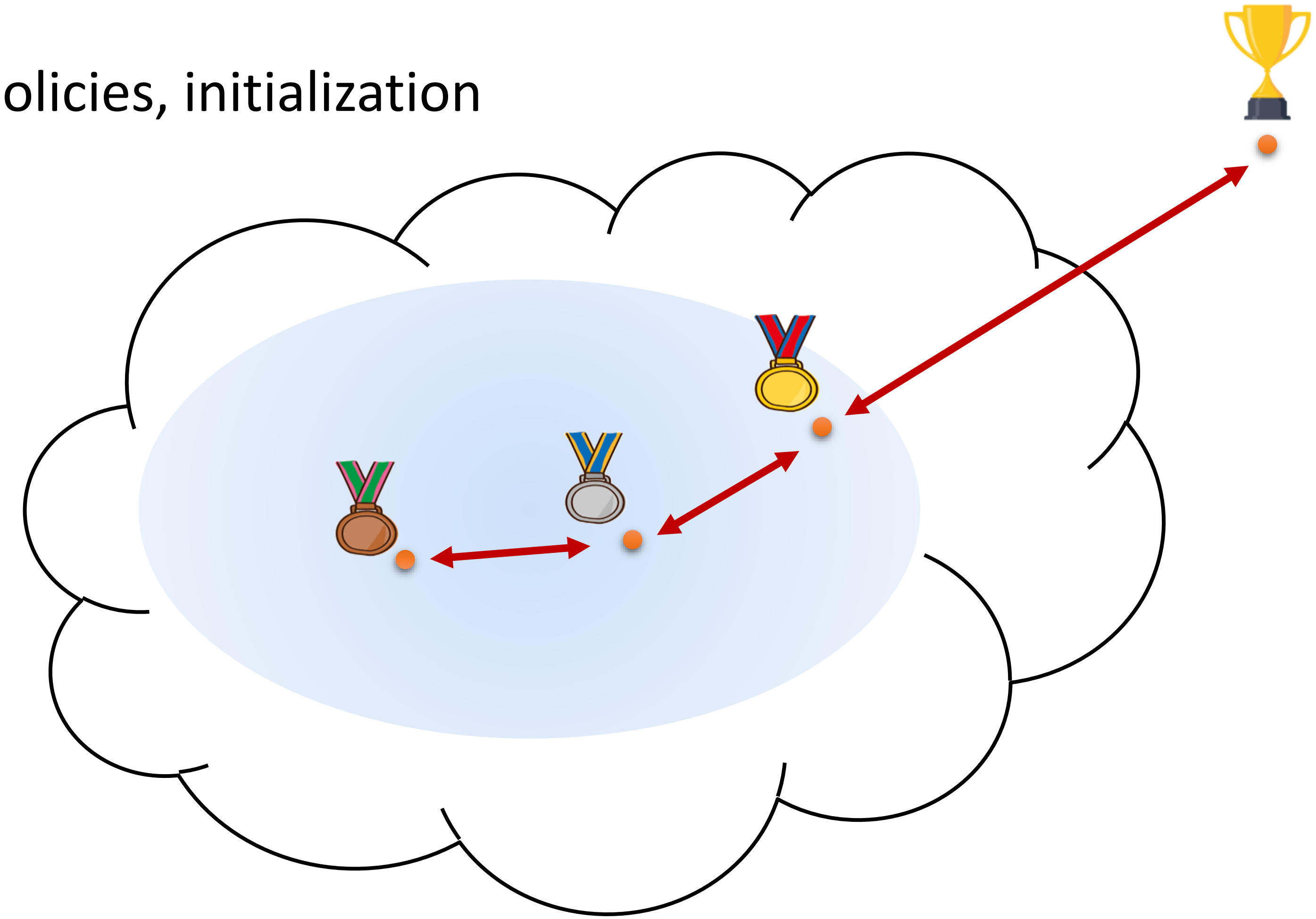
Optimization pitfalls

Too small learning rate



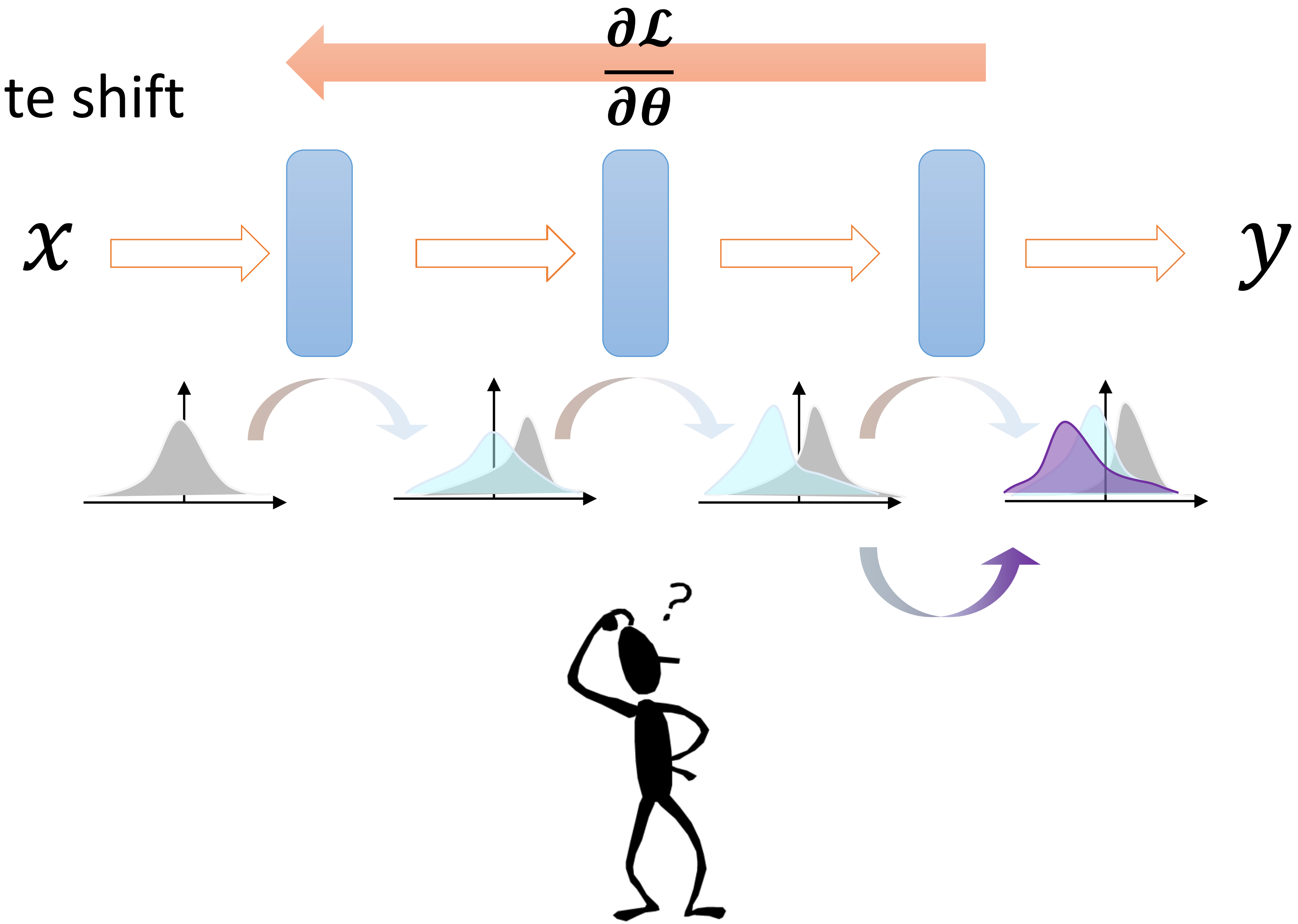
Agenda

- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization



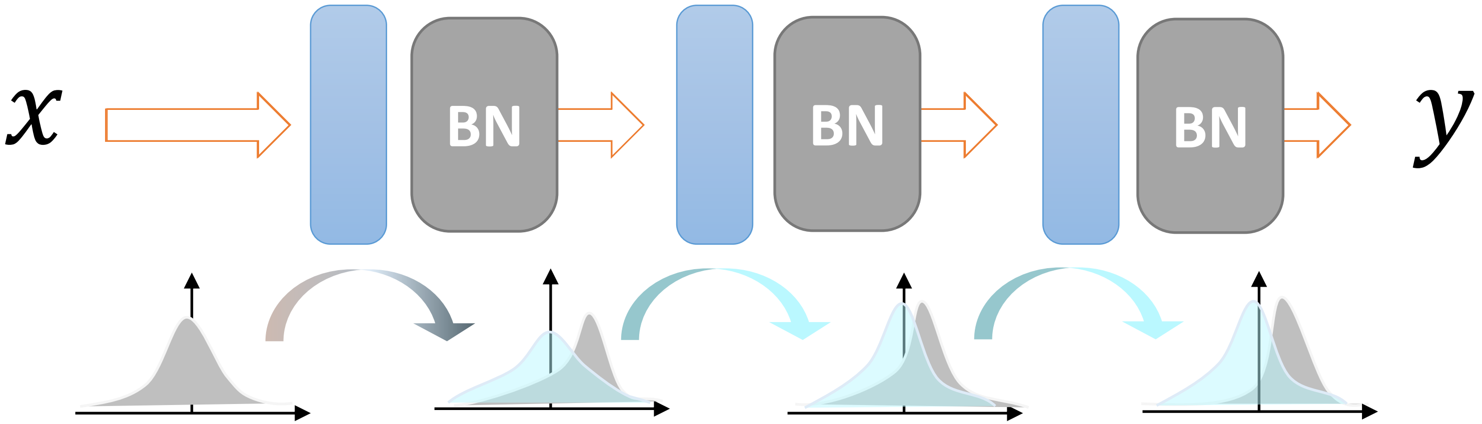
Optimization's Pitfalls

- Covariate shift



Optimization's Pitfalls

- Batch Norm!



Agenda

- SGD
  - Momentum, Adam, LR policies, initialization
- Regularization
  - DropOut
  - Weight Decay
  - Augmentation
  - Early stopping
- Batch normalization

What’s coming up…

- Tutorial – Advanced pytorch
- Lecture (next week) – Visualization and Understanding